UNIVERSITÉ FRANÇOIS - RABELAIS TOURS

UFR Sciences et Techniques
Licence S&T 1ère année

Informatique — UE 102

Architecture des ordinateurs et Algorithmique de base

Jean-Yves Antoine

http://www.intro.univ-tours.fr/~antoine/

UNIVERSITÉ FRANÇOIS - RABELAIS TOURS

UFR Sciences et Techniques

Licence S&T 1ère année

Informatique UE 102

Chap II – Programmation impérative : introduction

Où l'on voit que programmer est plus affaire d'analyse de problème que de codage

PROGRAMMATION ET GENIE LOGICIEL

Luttons contre quelques idées reçues

- ✓ Avant d'être une suite d'instructions, un programme constitue une réponse à un problème donné
- ✓ La spécification algorithmique du programme et son codage proprement dit de ce programme ne constituent donc qu'une petite partie du développement logiciel

  ⇨**cycle de vie** en **génie logiciel**

Pourtant … objectifs de ce cours

- ✓ Centré sur l'algorithmique et l'activité de programmation proprement dite

  ⇨**bases** avant de découvrir les problèmes réel du génie logiciel

PROGRAMMATION ET GENIE LOGICIEL

En entreprise : « programming in-the-large »

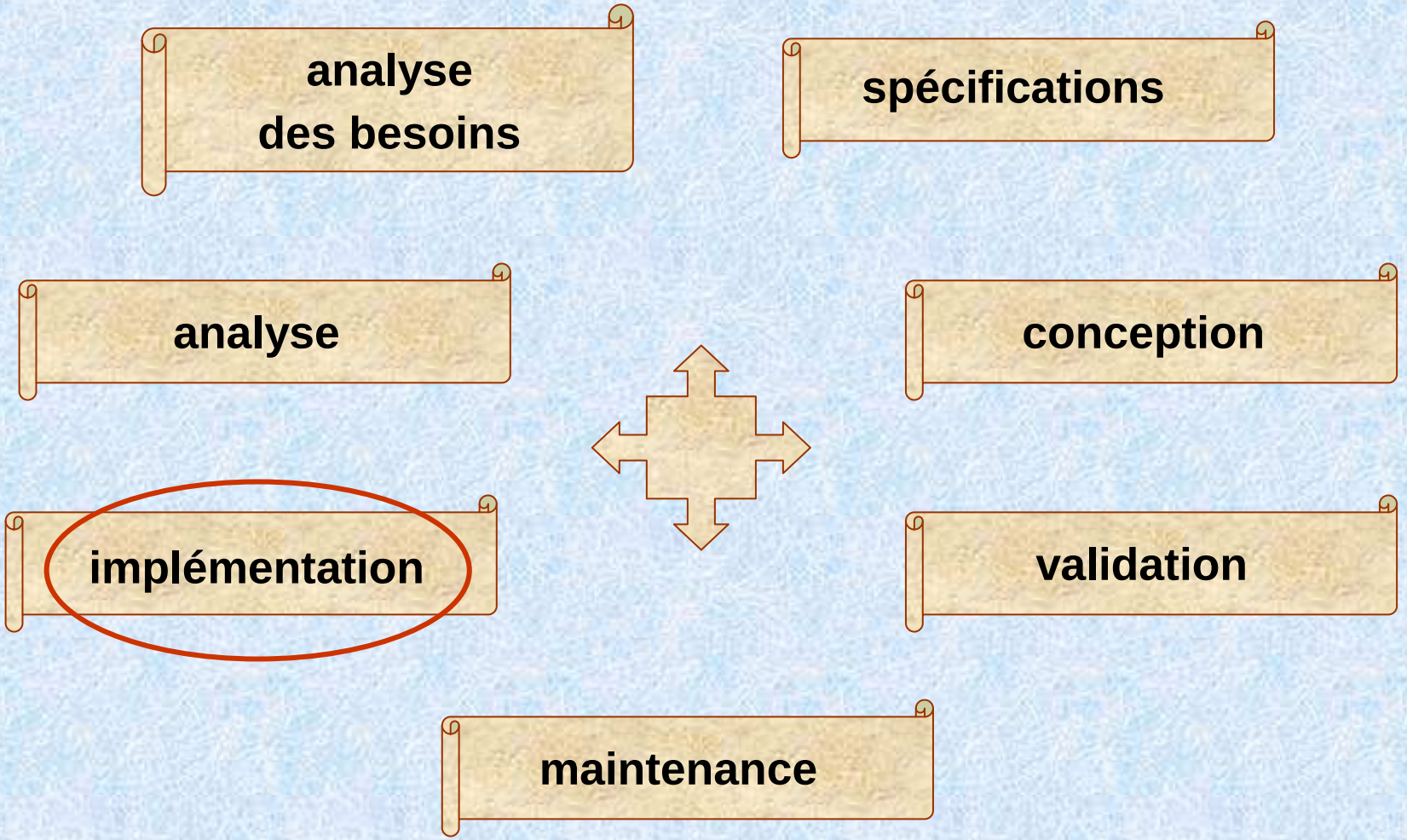
- ✓ Problème du client pas toujours bien spécifié
- ✓ Client pas nécessairement informaticien: parler métier
- ✓ Plusieurs dizaines voire centaines de milliers de lignes de code
- ✓ Travail en équipe sur des projets longs et complexes : méthodologie
- ✓ Importance de la qualité du logiciel

En licence 1 : « programming in-the-small »

- ✓ Problème spécifié précisément de manière formelle
- ✓ Au plus une centaine de ligne de code
- ✓ Travail seul ou en binôme
- ✓ La qualité du logiciel ne joue « que » sur votre note ☺

PROCESSUS DE DEVELOPPEMENT LOGICIEL

DES ETAPES MULTIPLES



PROCESSUS DE DEVELOPPEMENT LOGICIEL

ANALYSE DES BESOINS

Étape préalable si le client a une idée peu précise du système à réaliser

- Étude informelle des fonctionnalités (externes) du système sans considération technique : point de vue métier / utilisateur
- Dialogue fournisseur / client en termes intelligibles pour ce dernier : l'aider à formaliser le problème à résoudre
- Conduit à la définition du cahier des charges

Exemple : système d'accès à l'IUP

Le bâtiment de l'IUP se divise en 2 zones (enseignement, administration et enseignants) correspondant à des droits d 'accès différents. ***150*** *personnes ont quotidiennement accès au site. En dehors des visiteurs, tous les autres catégories de personnes disposeront d 'une carte d 'accès : enseignants et administratifs (15) et étudiants (135). Les visiteurs n 'auront accès au bâtiment qu'aux heures de libre ouverture (****voir planning****). En dehors de ce horaires, l'autorisation d'accès dans chaque zone sera limitée au porteurs de badge et dépendra de leur statut et de l 'horaire. Chaque zone est contrôlée par une porte d 'accès munie d'un lecteur de badge spécifique. Les droits d'accès ainsi que le planning des horaires d 'ouverture seront définis par un* ***superviseur*** *….*

PROCESSUS DE DEVELOPPEMENT LOGICIEL

SPECIFICATIONS

Définit ce que doit faire le système (côté client)

- Document précis spécifiant les fonctionnalités attendues
- Base du contrat commercial avec le client
- Document facile à comprendre par le client / utilisateur

Exemple — *Description des fonctionnalités du système avec scénarios, interactions : enchaînements d'écrans*

Use case : Configuration système

1) **identification**
   a) saisie du **code personnel**
   b)vérification du code personnel
   c) autorisation
2) Modification de configuration
   a) choix d 'une zone

PROCESSUS DE DEVELOPPEMENT LOGICIEL

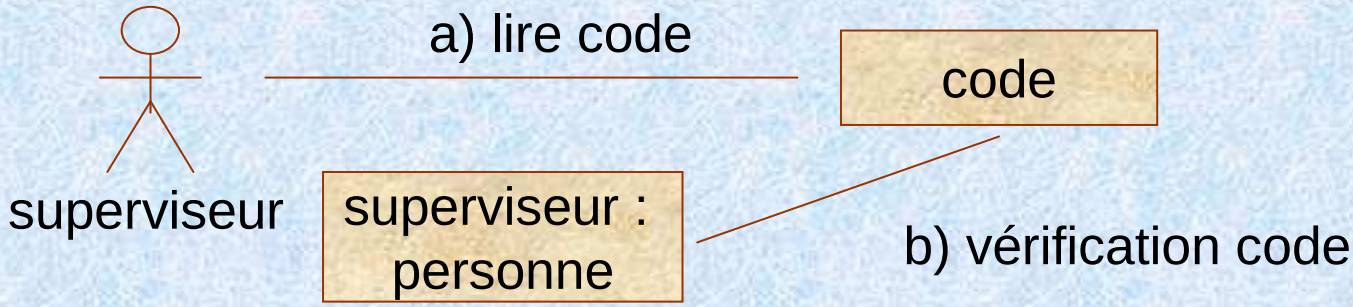
ANALYSE

« Quoi faire » : comprendre et modéliser le métier

- **Premier modèle du système (niveau métier)**

  Identifier les éléments intervenants hors (acteurs) et dans le système : fonctionnalités, structure et relations, états par lesquels ils passent suivant certains événements.
- Réflexion métier hors de toute considération technique
- Reste un support de discussion avec le client/utilisateur

Exemple : *diagrammes de classes, de collaborations...*



PROCESSUS DE DEVELOPPEMENT LOGICIEL

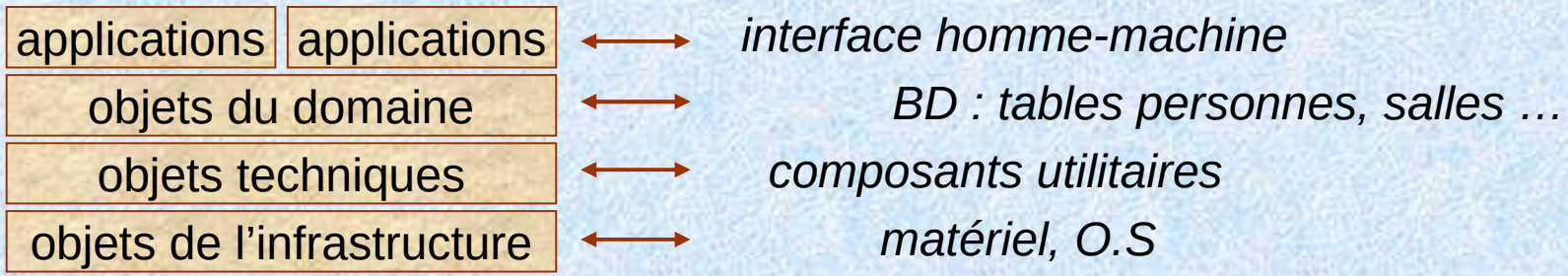
CONCEPTION

Comment faire le système : choix techniques

Modèle de l'architecture logicielle du système

- Décomposition en sous-systèmes : application (interfaces), domaine (métier) et infrastructure (implémentation).
- Choix de l'architecture technique suivant certaines priorités (robustesse, efficacité, portabilité…)
- Expertise informatique : hors de la compréhension du client

Exemple : *modèle d'architecture en couche*



PROCESSUS DE DEVELOPPEMENT LOGICIEL

IMPLEMENTATION

Activité de programmation à proprement parler

Souvent trop de temps consacré au codage au détriment des phases d'analyse et de conception : mauvaise pratique très coûteuse…

⇨ *Remarque — L'activité de développement sera de plus en plus tournée vers la réutilisation de composants existants*

VALIDATION

- Tests en **interne**
- **Recette** : validation client (pas seulement absence de bug mais surtout accord sur les besoins satisfaits : cahier de recette)

PROCESSUS DE DEVELOPPEMENT LOGICIEL

MAINTENANCE

- correction des erreurs du système ⇨**robustesse**
- demande d'évolution (modification de l'environnement technique, nouvelle fonctionnalités) ⇨**modifiabilité**, **portabilité**

Étape longue, critique et coûteuse

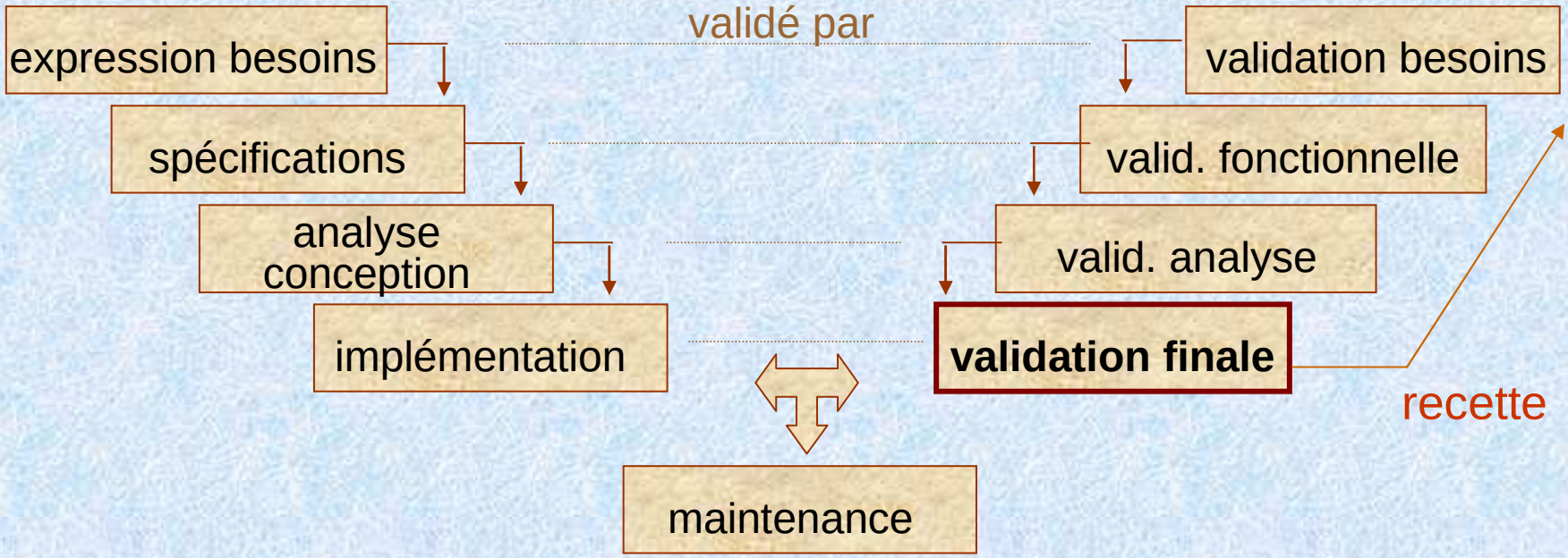
❶ Projet typique cité par [Mueller et Gaertner, 2000]

❷ Projet concernant 5 personnes, d'une durée de 12 mois avec maintenance longue (plusieurs années)

	❶	❷
✓ analyse des besoins et spécifications	15 %	6%
✓ analyse / conception	25%	5 %
✓ implémentation	30 %	7 %
✓ tests et validation	15%	15 %
✓ maintenance	15 %	67 %

CYCLE DE VIE DU LOGICIEL

CYCLE EN « V »

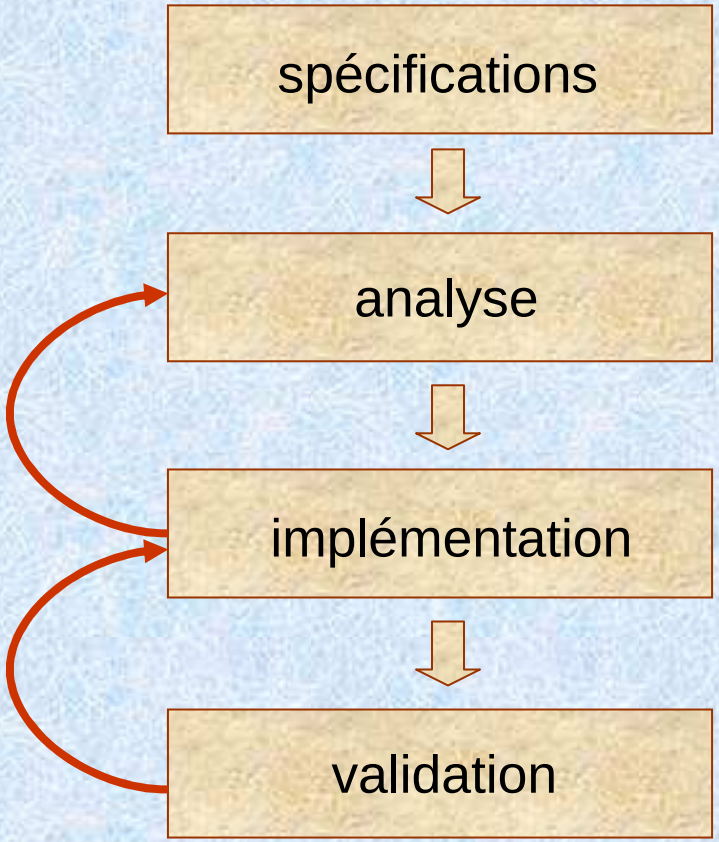


AUTRES METHODOLOGIES

- analyse en cascade
- cycle en spirale

PROGRAMMATION « IN-THE-SMALL »

- Spécifications — données, complètes et précises
- analyse descendante — décomposition du problème / algorithmes
- implémentation — un seul langage de programmation
- validation — jeux d'essais



DEVELOPPEMENT LOGICIEL ET METHODOLOGIE

- Le développement logiciel n'est pas une affaire de bidouille réservée à des initiés maîtrisant des « super » astuces
- Il nécessite au contraire une méthodologie rigoureuse qui le rapproche avant tout des mathématiques ⇨ **génie logiciel**

LA « CRISE » DU LOGICIEL

- Démarche ingéniérique encore mal intégrée par les informaticiens
- Pannes logicielles
  - avion F16 retourné au passage de l'équateur : non prise en compte du référentiel hémisphère sud
  - échec sonde Venus : virgule remplacée par un point
  - bogue de l'an 2000 ...
  - instabilité de Windows …

⇨ **Coût global** : 80 Mrd$ (1998) ; 175 Mrd$ (2000)

⇨ **Systèmes critiques** : nucléaire, transport ($\approx 10^{-10}$ pannes/h), système bancaire...

QUEL LANGAGE DE PROGRAMMATION ?

DIFFERENTS TYPES DE PROGRAMMATION

- **Impérative**
  - ⇨ Pascal, Fortran, Cobol, Basic, C, ADA
- **Fonctionnelle**
  - ⇨ Lisp, Scheme, CAML
- **Logique**
  - ⇨ Prolog
- **Distribuée (Objet)**
  - ⇨ Smalltalk, C++, JAVA

Ada Lovelace :
L'informatique n'est pas réservée qu'aux hommes…

DIFFERENTS TYPES DE PROGRAMMATION

PROGRAMMATION IMPERATIVE

Programmation **proche du fonctionnement itératif de l'ordinateur**

- instructions qui manipulent à la suite des informations de la mémoire, avec branchements conditionnels dans le programme comme ceux gérés par le compteur ordinal ⇨le programme dit « **comment faire** »
- efficace en terme de rapidité de calcul
- éloignée de l'analyse du problème (« quoi faire »)

Exemple — calcul de la factorielle d'un entier n

- Mettre 1 dans la variable *fact*
- Lire la valeur n au clavier
- POUR toutes les valeurs d'un compteur *i* variant de 1 jusqu'à n FAIRE
  multiplier *fact* par *i* : nouvelle valeur de *fact*.
- Afficher la valeur finale *fact*

DIFFERENTS TYPES DE PROGRAMMATION

PROGRAMMATION DECLARATIVE / SYMBOLIQUE

- Programmation fonctionnelle
- Programmation logique

- Programmation de plus haut niveau, proche du problème ⇨on dit « **quoi faire** » plutôt que « comment faire »
- Décomposition récursive du problème en sous-problèmes identiques mais plus simples
- Programmation facilitée mais fonctionnement moins efficace

Exemple — calcul de la factorielle d'un entier n

- La factorielle de 0 est égale à 1
- Pour tout n > 0, la factorielle de n est égale à la factorielle de (n-1) multipliée par n.

Remarque — *Programmation récursive possible avec les langages impératifs*

DIFFERENTS TYPES DE PROGRAMMATION

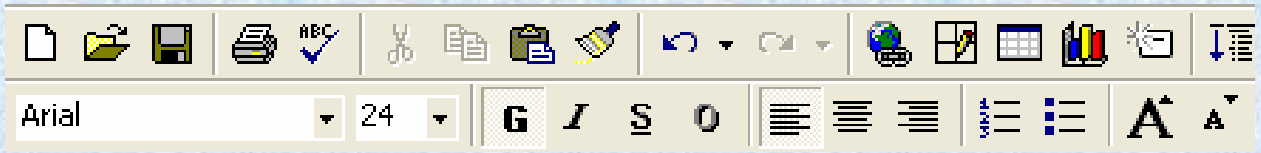
PROGRAMMATION DISTRIBUEE (OBJET)

Dimension transversale en terme de type de programmation (on peut faire de l'impératif objet ou du logique objet)

Chaque élément du programme correspond à une entité active et indépendante (objet ou composant) doté de ses propres ressources / propriétés. La résolution du problème passe par une communication entre les objets

- Méthodologie de modélisation essentielle (UML…)
- Intérêt : réutilisabilité des composants

Exemple — objets OLE



QUELQUES NOTIONS COMPLEMENTAIRES

DECIDABILITE et CALCULABILITE [A. Church]

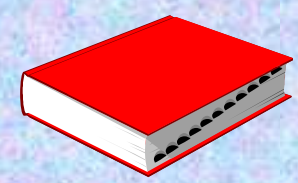
Il existe des problèmes non implantables sous la forme d'un programme (i.e problème non décidable)

Problèmes de terminaison : lorsque vous réaliser un programme récursif, si celui-ci présente une **boucle infinie** à l'exécution, la cause en sera certainement un codage erroné plutôt qu'un problème de non décidabilité…

COMPLEXITE complexité linéaire, polynomiale, exponentielle (NP)

CORRECTION DE PROGRAMMES

- ✓ En règle générale, il n'existe pas de méthode exhaustive pour tester complétement la correction d'un programme
- ✓ Langage B et consorts
- ✓ Méthodologie rigoureuse pour prévenir les erreurs et faire une validation aussi exhaustive que possible



Bibliographie

Ouvrages généraux

A faire…

Cours sur la Toile

Supports du cours : www.sir.blois.univ-tours.fr/~antoine/enseignement/pascal